Milad Mikeal

CS 162

10/7/18

Lab 3

Design:

- Create menu
  - Ask user to start the game or quit
- Create die object
  - Create roll method that selects a random number between 1 and N
    - N being the number of sides
- Create loaded die object
  - Inherit loaded die object from die object
  - Create roll method that overrides the inherited roll method
    - Loaded die can only roll top half numbers
- Create game object
  - Call menu
  - Prompt user to enter rules of the game
    - Rounds, whether each player will have a loaded die, and number of sides for each player's die
    - Validate each input
  - Select appropriate die for each player based on user's input
  - Set the number of sides for each die based on user's input
  - Start game
    - Loop until rounds played = rounds selected by user
      - Roll die for each player
      - Add one to points of player that had a higher roll
      - Increment rounds played
      - Print stats
    - Declare winner based on who had more points

Menu Test Table: 1 for start, 2 for quit

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 2	Prompt user to re-enter value	User prompted to re-enter value
Input in range	Input = 1 or Input = 2	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

Rounds Test Table:

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 21	Prompt user to re-enter value	User prompted to re-enter value
Input in range	Input between 1 - 21	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

Loaded Die Test Table: 1 for yes, 2 for no

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 1	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 2	Prompt user to re-enter value	User prompted to re-enter value
Input in range	Input = 1 or Input = 2	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

Number of Sides Test Table:

Test Case	**Input**	**Expected Outcome**	**Outcome**
Input too low	Input < 4	Prompt user to re-enter value	User prompted to re-enter value
Input too high	Input > 20	Prompt user to re-enter value	User prompted to re-enter value
Input in range	Input between 4 - 20	Proceed accordingly depending on user's input	Program proceeded accordingly
Non-numeric input	Input = alpha	Prompt user to re-enter value	User prompted to re-enter value

Reflection:

I did not think this lab was not too difficult. I actually enjoyed it. Once again, my original design was overcomplicating tasks. I was originally drawing my design to include Player objects, and then I realized it's not that necessary. I could just create everything within the game object, so I redesigned my program without Player objects before beginning. As a result, when I began implementing the game, there was not much I needed to change. My actual program was just translating my design to code.

I did have one hiccup along the way though. I seeded rand with time(0), and when the game would start, all the rolls got seeded with the same number. Therefore, all the rolls were resulting in a tie. So I had to find a way fix this, which I did by having the user hit enter to simulate a roll for each player, each turn. I also added a bit of code to have the system sleep for 1 second, therefore, each roll was not seeded with the same exact time. This helped solve the problem and added a bit more interaction to the game.